8048 program computes 16-by-8-bit quotient

by Donald W. McGuire
Red Lion Controls, York, Pa.

Many 16-by-8-bit divide routines for 8-bit microcontrollers yield an 8-bit result and an 8-bit remainder, thus forcing the divisor to be greater than the most significant byte of the dividend so that the result may be contained in 1 byte. But for many industrial applications, this restriction is intolerable.

The divide routine presented here for the 8048 microcontroller eliminates the above requirement by furnishing a 16-bit resultant and an 8-bit remainder. Yet it uses only the same number of registers as a typical 16-by-8-bit divide routine.

In the algorithm employed, the divisor is shifted left until the most significant bit is a 1 and the cycle counter, register R_5, is incremented once for each shift. Next the dividend is shifted left until the most significant bit of the high-order byte is a 1; the cycle counter is then decremented for each shift. The divisor is repeatedly subtracted from the higher-order byte of the dividend and the results shifted left. A 0 is added to the result if the subtraction generates a carry, a 1 if it does not do so. This process continues until the cycle counter is decremented to 0.

In order to facilitate simultaneous shifting of the dividend and the results, register R_2 and the accumulator are used to hold the result and the initial dividend. When R_5 reaches zero, the division process is complete and the remainder is separated from the quotient. The remainder, which is contained in the high-order byte of the results, is first brought to its proper significance by shifting it to the right until its least significant bit is located in bit 0 of the accumulator. □

8048 PROGRAM FOR 16-BY-8 BIT DIVISION

```
00009                    ;
00010                    ;
00011                    ;          THIS IS A 16-BY-8-BIT DIVIDE ROUTINE THAT
00012                    ;          PROVIDES A TWO-BYTE RESULT AND A ONE-
00013                    ;          BYTE REMAINDER
00014                    ;
00015                    ;          AT ENTRY:
00016                    ;          A  = UPPER 8 BITS OF DIVIDEND
00017                    ;          R2 = LOWER 8 BITS OF DIVIDEND
00018                    ;          R1 = POINTER TO DIVISOR IN RANDOM-ACCESS MEMORY
00019                    ;
00020                    ;          AT EXIT:
00021                    ;          A  = UPPER 8 BITS OF RESULT
00022                    ;          R2 = LOWER 8 BITS OF RESULT
00023                    ;          R1 = POINTER TO REMAINDER IN RAM
00024                    ;          C  = SET IF DIVISOR IS 0 OR
00025                    ;               GREATER THAN THE DIVIDEND
00026                    ;
00027                    ;
00028                    ;
00029                    ;
00030  0100  960A  DIVIDE    JNZ   DIVIDE1    ; THIS SECTION
00031  0102  2A              XCH   A, R2      ; TESTS THAT
00032  0103  37              CPL   A          ; THE DIVIDEND
00033  0104  61              ADD   A, @R1     ; IS GREATER
00034  0105  37              CPL   A          ; THAN THE
00035  0106  61              ADD   A, @R1     ; DIVISOR
00036  0107  F653            JC    ERROR1
00037  0109  2A              XCH   A, R2
00038  010A  21    DIVIDE1   XCH   A, @R1     ; TEST IF
00039  010B  C652            JZ    ERROR      ; DIVISOR "0"
00040  010D  BD01            MOV   R5, #01    ; PRELOAD COUNT
00041  010F  F215  DIVIDE2   JB7   DIVIDE3    ; LEFTHAND JUSTIFY
00042  0111  E7              RL    A          ; THE DIVISOR
00043  0112  1D              INC   R5         ; INCREASE CYCLE
00044  0113  240F            JMP   DIVIDE2    ; COUNT
```

```
00045  0115  19      DIVIDE3   INC   R1          ; STORE
00046  0116  2D                XCH   A, R5       ; COUNT
00047  0117  A1                MOV   @R1, A      ; IN RAM
00048  0118  0308              ADD   A, #08      ; INCREASE
00049  011A  2D                XCH   A, R5       ; COUNT BY 8
00050  011B  C9                DEC   R1
00051  011C  85                CLR   F0
00052  011D  21                XCH   A, @R1
00053  011E  F227    DIVIDE4   JB7   DIVIDE5     ; LEFTHAND
00054  0120  97                CLR   C           ; JUSTIFY THE
00055  0121  2A                XCH   A, R2       ; DIVIDEND
00056  0122  F7                RLC   A
00057  0123  2A                XCH   A, R2       ; DECREASE
00058  0124  F7                RLC   A           ; CYCLE
00059  0125  ED1E              DJNZ  R5, DIVIDE4 ; COUNT
00060  0127  37      DIVIDE5   CPL   A           ; SUBTRACT DIVISOR
00061  0128  61                ADD   A, @R1      ; FROM MOST SIGNIFICANT BYTE OF
00062  0129  37                CPL   A           ; DIVIDEND
00063  012A  E62F              JNC   DIVIDE6     ; IT
00064  012C  B631              JF0   DIVIDE7     ; FITS
00065  012E  61                ADD   A, @R1      ; DOES NOT FIT
00066  012F  A7      DIVIDE6   CPL   C           ; PREPARE FOR SHIFT
00067  0130  95                CPL   F0          ; INTO RESULT
00068  0131  2A      DIVIDE7   XCH   A, R2       ; SHIFT CARRY
00069  0132  F7                RLC   A           ; INTO RESULTS
00070  0133  2A                XCH   A, R2
00071  0134  F7                RLC   A
00072  0135  E638              JNC   DIVIDE8     ; 1 SHIFTED
00073  0137  85                CLR   F0          ; INTO CARRY
00074  0138  95      DIVIDE8   CPL   F0          ; SET FLAG
00075  0139  ED27              DJNZ  R5, DIVIDE5 ; REPEAT
00076  013B  A1                MOV   @R1, A      ; SAVE HIGH RESULT
00077  013C  19                INC   R1          ; GET
00078  013D  F1                MOV   A, @R1      ; INITIAL
00079  013E  C9                DEC   R1          ; COUNT
00080  013F  AD                MOV   R5, A
00081  0140  F1                MOV   A, @R1      ; GET HIGH RESULT
00082  0141  67      DIVIDE9   RRC   A
00083  0142  97                CLR   C           ; RESTORE
00084  0143  ED41              DJNZ  R5, DIVIDE9 ; REMAINDER
00085  0145  21                XCH   A, @R1      ; SAVE IT
00086  0146  19                INC   R1          ; GET
00087  0147  21                XCH   A, @R1      ; INITIAL
00088  0148  2D                XCH   A, R5       ; COUNT
00089  0149  97      DIVIDEA   CLR   C
00090  014A  A7                CPL   C           ; REMOVE
00091  014B  F7                RLC   A           ; REMAINDER
00092  014C  ED49              DJNZ  R5, DIVIDEA ; FROM
00093  014E  51                ANL   A, @R1      ; HIGH RESULT
00094  014F  C9                DEC   R1
00095  0150  97                CLR   C
00096  0151  83                RET
00097  0152  21      ERROR     XCH   A, @R1      ; SWAP DIVISOR/DIVIDEND
00098  0153  97      ERROR1    CLR   C
00099  0154  A7                CPL   C           ; SET CARRY
00100  0155  83                RET
00101                   END
```